DEVSUPERIOR

Formação Desenvolvedor Moderno
Módulo: Front end

Capítulo: Formulários, CRUD completo

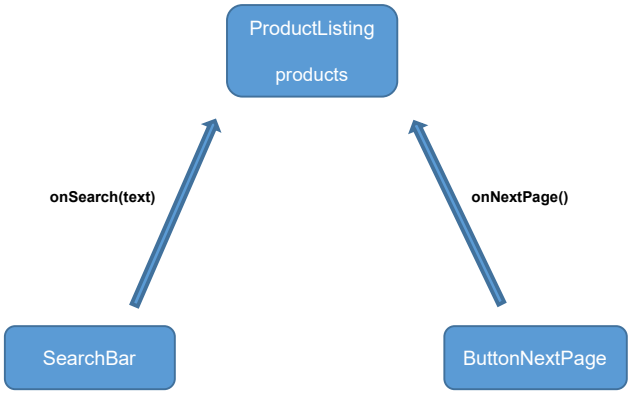
https://devsuperior.com.br

Tópicos

- (R) Tabela de dados paginados
- (D) Dialog modal
  - Aviso
  - Confirmação
- (U, C) Formulários
  - Discussão: biblioteca vs. implementação manual
  - Validação no front end
  - Regras de validação, expressões regulares
  - Exibição de mensagem para cada campo
  - Comportamento "sujo"
  - Exibição de mensagem global
  - Integração com controles de terceiros (Select)
  - Validação do back end

Tabela de dados paginados

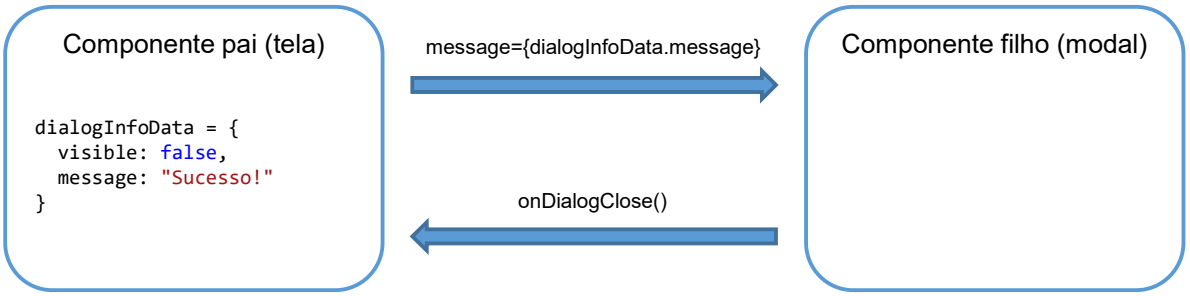
https://github.com/devsuperior/dscommerce-html-css



3

3

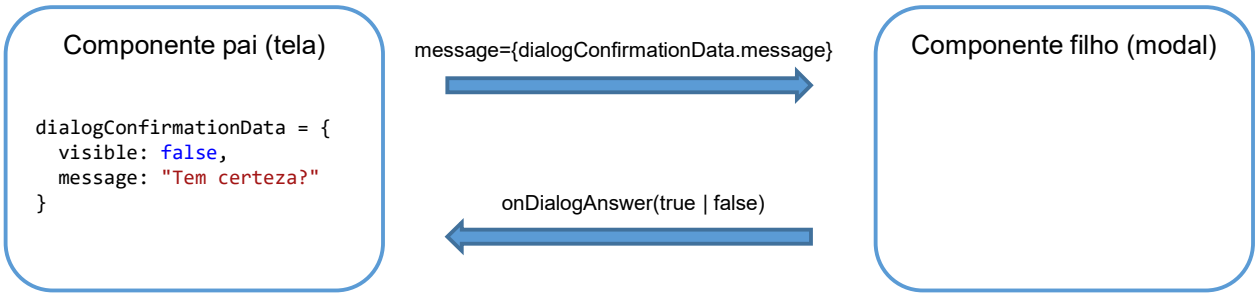
Dialog modal de aviso (DialogInfo)



4

4

Dialog modal de confirmação (DialogConfirmation)



Formulários: biblioteca vs. implementação manual

- Bibliotecas
  - React Hook Form
  - Formik
- Vantagens
  - Projeto maduro
  - Muitas features
  - Muitos usuários e devs
- Desvantagens
  - Vendor lock-in
  - Solução muito grande para problemas pequenos
  - Customizações devem aderir ao modelo / curva de aprendizagem

Formulários : biblioteca vs. implementação manual

Implementação manual

- Vantagens
  - Controle/domínio total do código
  - Solução do tamanho da necessidade
- Desvantagens
  - Custo para implementar tudo do zero
  - Requer habilidade de design

Estratégia para controle de formulário

O que queremos?

- Validação no front end
- Regras de validação, expressões regulares
- Exibição de mensagem para cada campo
- Comportamento "sujo"
- Exibição de mensagem global
- Integração com controles de terceiros (Select)
- Validação do back end

Estratégia para controle de formulário

Estado com os dados formulário

```
const [formData, setFormData] = useState<any>({
    description: {
        value: "Computador Gamer",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 2000.0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
             return value > 0;
        }
        message: "Informe um preço positivo",
    },
})
```

forms.ts
Funções que transformam os dados do formulário

Componentes customizados para inputs

```
<FormInput
    {...formData.description}
    className="dsc-form-control"
    onChange={handleChange}
    onTurnDirty={handleTurnDirty}
/>
```

Dataset nos inputs
`data-dirty="true" data-invalid="false"`

CSS
Estilos para inputs e erros do formulário

Formato dos dados do formulário

```
const [formData, setFormData] = useState<any>({
    username: {
        value: "",
        id: "username",
        name: "username",
        type: "text",
        placeholder: "Email",
        validation: function (value: string) {
            return /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/.test(value.toLowerCase());
        },
        message: "Favor informar um email válido",
    },
    password: {
        value: "",
        id: "password",
        name: "password",
        type: "password",
        placeholder: "Senha",
    }
})
```

Componente FormInput básico

```
export default function FormInput(props: any) {

    const { ...inputProps } = props;

    return (
        <input {...inputProps} />
    );
};
```


Função update(inputs: any, name: string, newValue: any)

Objetivo: gerar um novo objeto de formulário onde o campo de nome "name" seja atualizado com o valor "newValue"

Exemplo: `update(inputs, "description", "Computado")`

```
const inputs = {
    description: {
        value: "Computad",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

```
{
    description: {
        value: "Computado",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

Função toValues(inputs: any)

Objetivo: gerar um objeto contendo apenas os valores dos campos do formulário

Exemplo: `toValues(inputs)`

```
const inputs = {
    description: {
        value: "Computador Gamer",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 2000.0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

```
{
    description: "Computador Gamer",
    price: 2000.0
}
```


Função updateAll(inputs: any, newValues: any)

Objetivo: gerar um novo objeto de formulário onde os campos sejam os valores contidos em "newValues"

Exemplo: `updateAll(inputs, {description: "Computador", price: 2000.0} )`

```
const inputs = {
    description: {
        value: "",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

```
{
    description: {
        value: "Computador",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 2000.0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

Função validate(inputs: any, name: string)

Objetivo: gerar um novo objeto de formulário contendo o campo "invalid" (que pode valer "true" ou "false") para o campo de formulário cujo nome é "name"

Exemplo: `validate(inputs, "price")`

```
const inputs = {
    description: {
        value: "Computador Gamer",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

```
{
    description: {
        value: "Computador Gamer",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
        invalid: "true"
    },
}
```

15

15

CSS para mostrar os erros

```
.dsc-form-error {
  color: var(--dsc-color-error);
  font-size: 12px;
  padding-left: 4px;
  display: none;
}

.dsc-form-control[data-invalid="true"] {
  border: 1px solid var(--dsc-color-error);
}

.dsc-form-control[data-invalid="true"] ~ div {
  display: unset;
}
```

16

16

Função toDirty(inputs: any, name: string)

Objetivo: gerar um novo objeto de formulário contendo o campo "dirty" (que pode valer "true" ou "false") para o campo de formulário cujo nome é "name"

Exemplo: `toDirty(inputs, "price")`

```
const inputs = {
    description: {
        value: "Computador Gamer",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
    },
}
```

```
{
    description: {
        value: "Computador Gamer",
        id: "description",
        name: " description ",
        type: "text",
        placeholder: "Descrição"
    },
    price: {
        value: 0,
        id: "price",
        name: " price ",
        type: "number",
        placeholder: "Preço",
        validation: function (value: number) {
              return value > 0;
        }
        message: "Informe um preço positivo",
        dirty: "true"
    },
}
```

17

17

React Select

https://react-select.com

```
yarn add react-select@5.6.0
```

18

18

```
//https://stackoverflow.com/questions/52614304/react-select-remove-focus-border
//https://stackoverflow.com/questions/58801252/possible-to-change-font-color-on-react-select
//https://react-select.com/styles

export const selectStyles = {
  control: (provided: any) => ({
    ...provided,
    minHeight: "40px",
    border: "none",
    boxShadow: "none",
    "&:hover": {
      border: "none",
    },
  }),
  placeholder: (provided: any) => ({
    ...provided,
    color: "var(--dsc-color-font-placeholder)",
  }),
  option: (provided: any) => ({
    ...provided,
    color: "var(--dsc-color-font-primary)",
  }),
  indicatorSeparator: (provided: any) => ({
    ...provided,
    display: "none",
  }),
};
```


19